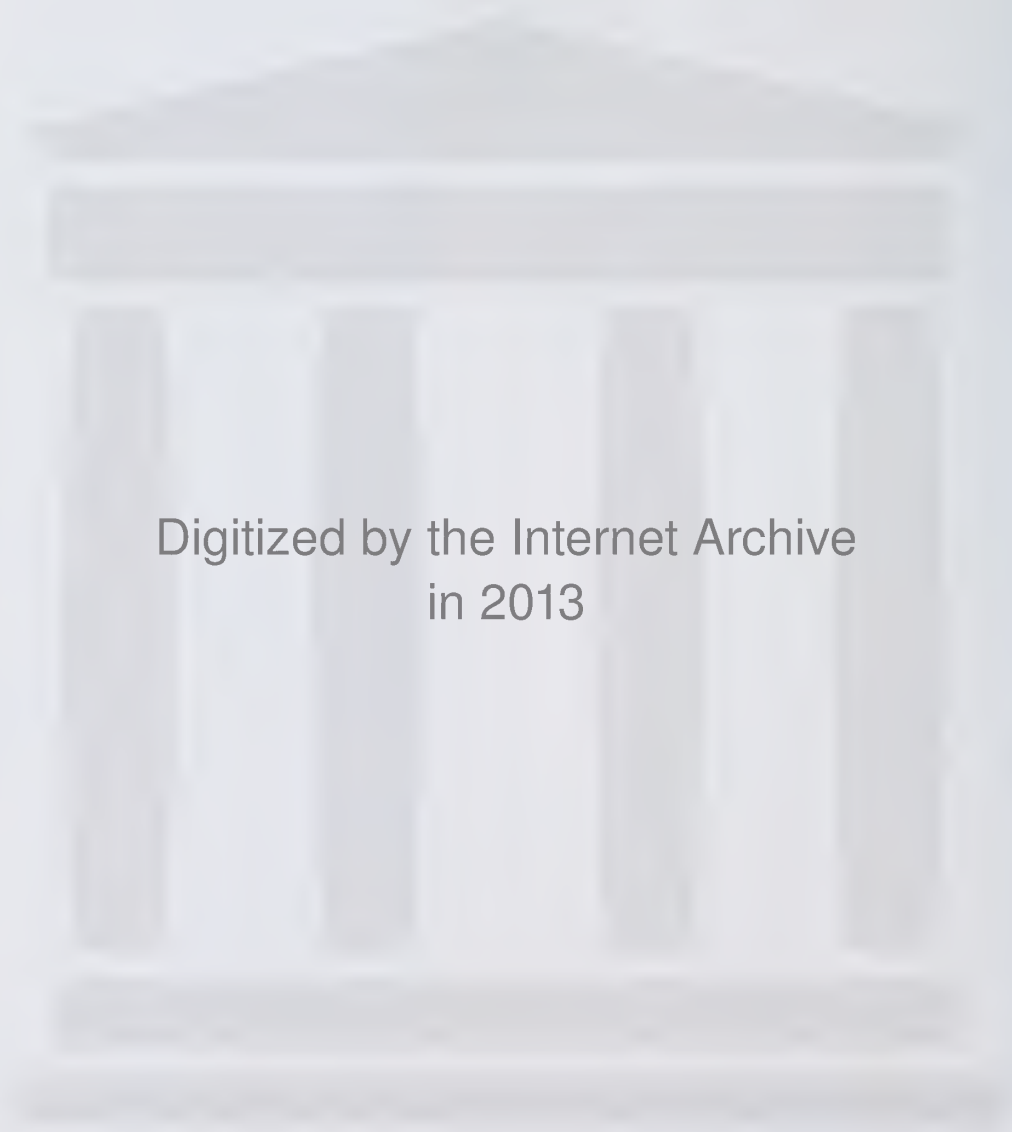

Report No. 317

COO-1469-0114

USER PROGRAMMING SPECIFICATIONS FOR THE PDP8/360 COMMUNICATIONS SYSTEM

by

M. J. Michel

April 1969

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

Report No. 317

USER PROGRAMMING SPECIFICATIONS FOR THE PDP8/360 COMMUNICATIONS SYSTEM*

by

M. J. Michel

April 1969

Department of Computer Science
University of Illinois
Urbana, Illinois 61801

* This report supported in part by Grant U. S. AEC AT(11-1)1469.

TABLE OF CONTENTS

1. INTRODUCTION

The following pages are not intended to be a definitive paper on the organization, use, or operation of the PDP8/360 communications system. They are intended only to provide some basic information essential to writing programs to run under the system and to give some idea of the facilities available.

2. BACKGROUND INFORMATION FOR 360/50-PDP7/PDP8 COMMUNICATION

A. PDP8 to PDP7 to 360/50

The PDP8 transmits or receives one character at a time to or from the PDP7 at a maximum rate of 100 characters per second. As soon as the PDP7 senses a carriage-return character from the PDP8 or when it has received a total of 125 characters without getting a carriage-return character, the PDP7 considers the "line" of characters from the PDP8 to be completed. At this time the PDP7 transmits a carriage-return followed by a line feed to the PDP8. The PDP7 then tries to send the line to the 360/50. When the 360/50 reads the line, and not before, the PDP7 sends a bell-character to the PDP8 signifying that the PDP8 may send another line. If the PDP8 should try to send characters to the PDP7 before receiving the bell-character, the PDP7 immediately sends a carriage-return line feed, "#WAIT", carriage-return, line feed sequence to the PDP8.

The format of the line sent to the 360/50 from the PDP7 is as follows:

```
A,B[125 data bytes]CR
0 1 2          126 127
```

Where A and B (bytes 0 and 1) are PDP7-360/50 control characters, byte 127 is always a carriage-return, and bytes 2 through 126 are data bytes. If the PDP8 had sent 50 characters followed by a carriage-return, the line would look like:

```
A,B[......CR---Junk---]CR
0 1 2   51 52       126 127
```

B. 360/50 to PDP7 to PDP8

When the 360/50 sends data for the PDP8 to the PDP7, the data format is:

```
AB[N---124 data bytes]CR
01 23            126 127
```

where A and B are PDP7-360/50 control characters, N is a control character for the PDP8 monitor system, and byte 127 is always a carriage-return. If 50 characters were to be sent to the PDP8, the line would be:

```
AB[N      CR---Junk---]C R
01 23 52 53 54       126 127
```

The PDP7 would send only bytes 2 through 53 to the PDP8, followed by an additional line feed (NOTE: When the PDP8 initiated a send, a line feed and a bell were returned; when the /50 initiates a send, only a line feed is additionally sent to the PDP8.).

After sending one line, the /50 cannot send another line until the first has been completely sent (character by character) to the PDP8. Note that the PDP7 assumes the PDP8 is a synchronous device; hence, the PDP7 sends characters to the PDP8 at a fixed rate and the PDP8 must accept them at that rate. In order to know when this has been accomplished, the program in the /50 must issue a READ operation. In this case, byte 1 ('B') of the input indicates the disposition of the last line. (On ordinary input, B = 0, meaning the line is a data line.) The rest of the line is junk. If B is

1, then the output has all been transmitted to the PDP8.

or 2, then the PDP8 has requested the PDP7 to stop transmission before the entire line was sent, i.e. the remainder of the line has been voluntarily lost;

or 3, then the PDP8 has "signed off."

Hence, to send several lines to the PDP8, the program in the /50 must issue a read before every output operation (except the first) and check B for the proper setting.

C. Data Format

Since certain characters, such as carriage-return and line feed cause the PDP7 to take some action, data--particularly object code--cannot be sent 8 bits at a time from the PDP8; nor is 8 bits really convenient considering the 12 bit word of the PDP8. In addition, the 360/50 cannot send data in its original form to the PDP7 either, since an inadvertant

carriage return in the middle of the line would cause the PDP7 to stop transmitting to the PDP8. Hence, all data (except pure character data sent only to the time sharing system) is sent in a coded format, six data bits (half of a PDP8 word) per 8 bit character at a time. Since internal PDP8 characters are 6 bits and the transmitted form takes up 8 bits (1 byte), conversion in the 360/60 to EBCDIC is very neat and clean. Numeric data is reformed into 360/50 half words, i.e. one PDP8 word of 12 bits becomes 2 bytes. In the PDP8 each incoming data byte is stripped of the excess 2 bits and every 2 characters are packed into one PDP8 word.

3. GRAPHIC MONITOR SYSTEM
OVERVIEW AND USER DESCRIPTION

A. Overview

The PDP8-360/50 monitor systems and 360/50 I/O macros have been designed to help reduce the bother and complication inherent in the 360/50-PDP7-PDP8 communication linkage. The package consists of communication and debugging structures embedded in a double ended monitor system, one end in a non-terminating partition in the 360/50 and the other end permanently resident in bank 3 to the PDP8.

At the highest level, a user sitting at the PDP8 teletype uses the PDP8 monitor to communicate with the 360/50 monitor. On command, the PDP8 monitor enters one of several modes: it can become a simple console plugged into the time sharing system, or it can command the 360/50 to load a user module into the 360/50 graphics partition or to load data into PDP8 core, etc. The following types of operations are done at this level.

1. Using the time sharing mode, the user can create symbolic source and data files, enter jobs into the regular ASP job stream, etc. Output from these operations is transmitted back to the PDP8 and displayed on the 338 scope for easy viewing. Concurrently, the output can be routed to dectape, to disk (PDP8 or 2314) or 1403 printer for hardcopy version.

2. Using monitor-to-monitor communication, the user can call non-interactive, service programs into the graphics partition. As an example, PDP8ASM (a 360 program for assembly of PDP8 source code) could be called in to assemble a PDP8 program from a source file created by use of the time sharing facilities. The object code is stored on 2314 disks, the lisitng is sent to 2314 disk for "off line" printing and/or to 338 scope for immediate viewing (instant turn-around). The user notes his assembly errors, calls time sharing, changes some statements in the source file, recalls PDP8ASM to try again (the first object file and source listing can be overwritten thus eliminating un-needed printout, et.al.). Depending on partition size the user could call in <u>any</u> program, enabling him to have instant turn-around, for example, for any language

on the system, or for any analysis program, etc. The user can then command the 360/50 monitor to pass files to the PDP8 (source, object code, data, et. al.). These can be added to the PDP8 disk file or tape file (using dectape).

3. Interactive programs can also be called into the graphics partition (this was the main intent all along). The usual sequence would be something like: assemble and perfect programs for the 360/50 and for the PDP8 in 1 and 2 above. Load the PDP8 programs into the PDP8 and add to dectape. Call in the new module just created into the 360/50, call in the new PDP8 programs. We now have user application programs running at both ends of the system.

Quite a different level of operation is now in effect. Two application programs are communicating with each other and with the user. The 360/50 programs use the same I/O macros used by the 360/50 monitor. The PDP8 programs use the I/O routines of the PDP8 monitor; namely, 'loader' and 'sender', for communication with the 360/50 and a system message 'SYSMSG' routine for communication with the user. The PDP8 monitor maintains control of the interrupt structure, sending control to user and system routines as appropriate. In particular strict control is maintained over PDP8-360/50 communication; namely, PDP7 messages and 360/50 system messages are routed to a special buffer for display to the user. When these occur, certain functions such as transmission from the PDP8 to the PDP7 are inhibited until the user takes corrective action. This may include waiting, retrying the operation, cancelling the operation, specifying new parameters, adding parameters, etc. All PDP8 system routines are easily accessible to user routines via calling sequences or parameter settings. The user can get back to the monitor system at any time, either from his program or by a manual interrupt.

If a user bomb-out occurs in the 360/50, control is automatically passed to a resident debugging routine. This routine communicates with a special section of the PDP8 monitor and is completely transparent to a user program in the PDP8. In brief, the package allows the user to "scroll" around through 360/50 core to pointer chase, to take "snaps", to patch 360/50 core, and to restart his 360/50 program, all from the PDP8/338.

B. System Description

The following macros are available to 360/50 programs for communication with the PDP8 via the PDP7:

PDP7IN	- call PDP7EXCP to input data
PDP7OUT	- call PDP7EXCP to output data
PDP7TR	- call PDP7EXCP subroutine for data translation
PDP7EXCP	- perform I/O operations
nDC	- assembly time macro for defining data and character strings

All OS/360 linkage conventions apply: Reg. 0, 1, 13, 14, 15 used by I/O routines.

MACRO CALL:

label PDP7IN AREA=label,LEN=m,MSG=label,ERR=label,SOP=label,EXT=Y

Description:

This macro causes the next input record from the PDP7 to be put into core at the location specified by the AREA operand. The amount of data passed to the user is equal to the (HEX) number of bytes specified by the LEN parameter plus one, the last character being a carriage-return (X'8D"). If 80 column card images were expected, LEN=80 should be coded and 81 characters would be transferred. (When scanning an input string, the user need only scan for the carriage-return rather than keep a character count. Also if the string were shorter than expected, a carriage-return from the PDP8 will appear earlier in the record indicating the end of the line. Of course, if the record length is unknown, a request for a maximum length record is best.) LEN must be $\leq$ 125. Both AREA and LEN may be specified as residing in registers, e.g. AREA=(5) means the address of the input area is in register 5. LEN=(6) means the length of the input request is in register 6.

The input operation can have three possible results: success with data received, success with a system code received, or EXCP channel-error (very rare). Three corresponding parameters may be coded. For any not coded control passes to the next sequential statement after the calling sequence.

SOP=lab	- control passes to 'lab' if data successfully received
MSG=lab	- control passes to 'lab' if a system message is received. In this case, bytes 0 and 1 (A and B) instead of data bytes are placed in the first two bytes of the user's input area for examination.
ERR=lab	- control passes to 'lab' if channel error occurs. Best action is to assume line lost and indicate same to user with sysmessage facility. Hence, effect retry.

If EXT=Y is coded, the appropriate linkage for an externally defined PDP7EXCP CSECT will be generated (c.f. PDP7EXCP macro).

MACRO CALL:

label PDP7OUT AREA=label,LEN=m,ERR=label,SOP=label,EXT=Y,SYS=nn,MON=nn

Description:

This macro causes the output record at the core location specified by the AREA operand to be sent to the PDP7. The amount of data passed to the PDP7 is equal to the (HEX) number of bytes specified by the LEN parameter. LEN must be $\leq$ 124. Both AREA and LEN may be specified as residing in registers (e.g. PDP7IN).

The output operation can have two possible results analogous to the read operation: success with data sent, or EXCP channel-error (very rare). (c.f. PDP7IN for a description of ERR and SOP.) The action of EXT=Y is identical to the PDP7IN case.

The data line sent to the PDP7 is formatted:

```
AB[NX...X CR--Junk--]C R
01 23   k k+1 k+2 126 127
```

where A and B are PDP7-360/50 system communication bytes N is the PDP8 graphics monitor byte, bytes 3 through k are the data bytes, byte k+1 is a carriage-return (X'8D') inserted by PDP7OUT, bytes k+2 through 126 are junk from previous operations (these are ignored by the PDP7), and byte 127 is a mandatory carriage-return (X'8D') inserted by PDP7OUT. If 124 data bytes had been specified, there would be no 'junk' bytes, and the carriage return at bytes k+1 and 127 would coincide at byte 127.

The SYS and MON operands are used to specify bytes B and N.

For SYS:

00 = Time Sharing OFF
01 = Time Sharing ON
02 = Log In
03 = Log Out
04 = Data Line

For MON:

84 = Input Data
A3 = System Message
80-83, 85-86 = System Options

NOTE: User default options are SYS = 04, MON = 84.

The user may specify MON = A3 for transmitting lines to the system message device. All other codes are for systems maintenance and are password protected at this time.

MACRO CALL:

```
PDP7EXCP  EXT=Y,DDNAME=PDP7DD
```

Description:

This macro actually performs input/output operations with the PDP7 when called via PDP7IN and PDP7OUT. PDP7EXCP must be present if either PDP7IN or PDP7OUT is used, unless EXT=Y is coded for PDP7IN and PDP7OUT. Then PDP7EXCP with EXT=Y must appear in another CSECT in the same load module. The DDNAME operand specifies the name of the DD card that indicates UNIT=PDP7 to the operating system, with the default DDNAME being PDP7DD.

PDP7EXCP should be the last statement in an assembly or if it is not, any statements that come after must be named as a new CSECT.

MACRO CALL:

label PDP7TR n,LAB=label,LEN=m,EXT={X,Y}

Description:

Depending on the first operand, n, this macro translates blocks of data ($\leq$ 256 characters) at run time from one format into another. The LAB operand specifies the location of the block to be translated, and LEN specifies its length in (HEX) bytes. Both LAB and LEN may be specified as registers (c.f. PDP7IN). On the first occurrance of a call for each type of translation where the EXT operand is not coded, the translation table and/or routine is generated in addition to the calling sequence. Each subsequent call for a particular translation type results in only the calling sequence being generated. If EXT=Y is coded, only the calling sequence is generated and the translation table and/or routine is assumed to be externally defined. If EXT=X is coded, only the teranslation table and/or routine is generated and calls are assumed to be externally defined. (NOTE: PDP7EXCP and all the translation tables and/or routines could be put in one central 'input/output' CSECT. Other CSECTS would only need to use the calling sequences, PDP7IN, PDP7OUT, and PDP7TR.

There are eight translation types defined at present:

n = 1	- translates from EBCDIC to ASCII
2	- translates from EBCDIC to coded (6 bit) ASCII
3	- translates from EBCDIC to coded (6 bit) BCD
4*	- translates from /360 half words (right 12 bits) to pairs of coded (6 bit) binary
5	- ASCII to EBCDIC
6	- coded ASCII to EBCDIC
7	- coded BCD to EBCDIC
8*	- coded binary pairs to /360 half words

For n = 1, 2, 3, 5, 6, or 7 carriage-return (X'8D') and line feed (X'8A') are preserved by the translation. For example, if X'8A' is the last byte in an EBCDIC string being converted to 6 bit coded ASCII (n = 2), the corresponding byte in the resultant string is still X'8A'.

* LEN is the number of half words to be transmitted.

MACRO CALL:

label nDC 'message',{OWN=}

Description:

This macro allows data definition at assembly time of ASCII, coded ASCII, coded BCD and octal data, depending on the character n.

Forms:

*n = A	- define the EBCDIC characters enclosed in apostrophes as ASCII character data (i.e. lab ADC 'IJK' becomes lab DC X'C9CACB')
*n = B	- define the EBCDIC character enclosed in apostrophes as coded ASCII character data (i.e. lab BDC 'ABC' becomes lab DC X'828486')
*n = C	- define the EBCDIC characters enclosed in apostrophes as coded BCD character data (i.e. lab 'ABC' becomes lab DC X'E2E4E6')
n = Ø	- define the octal numerals as four digit octal numbers in 6 bit coded binary (i.e. lab ØDC 173 becomes DC X'82F6')
n = DØ	- define the decimal number as one four digit octal number in 6 bit coded binary (i.e. lab DODC 123 becomes lab DC X'82F6')

* " or && used to define ' or & may <u>not</u> be split into two card images.

The OWN operand is valid with n = A, B, C, only and can be used to redefine the output character set of an nDC definition. The OWN operand specifies: 1) that an internal redefinition is being performed (and not a data definition) and 2) the number of characters in the message that replace each character in the output character set. For example, ADC 'C1010202',OWN=2 would cause the output equivalent of B, C, and D for subsequent ADC definitions to be changed from C2, C3, and C4 to 01, 02, and 02, while keeping C1 for A and all the other character definitions as previously defined as well. (i.e. before ADC 'ABCD' became DC X'C1C2C3C5'; now, ADC 'ABCD' becomes DC X'C1010203').

Also, the user may change the standard character set if he desires by use of the STANTAB L, message macro. For example, STANTAB 1, '345' would change A, B, and C to 3, 4, and 5. (i.e. formerly ADC 'ABCD' became DC X'C1C2C3C4' now ADC '345D' becomes DC X'C1C2C3C4'). Hence, the user may completely alter the character definition facilities if so desired. (NOTE: 1) any character set may not exceed 150 items. 2) replacement by use of OWN is character by character beginning with the first character group in the OWN 'message' replacing the first character result in the output set and continuing sequentially only until the 'message' is exhausted. Hence, resultant groups in the output set not affected by the OWN 'message' remain as previously defined. Users must exercise care in changing the character sets to insure that mixed length definitions within a set are not created. 3) error checking is difficult and expensive for this macro and is not extensive: users are urged to be very careful and follow procedures for its use closely.).

The following user-oriented routines have been provided by the UIMON8 monitor system to make communication between PDP8 programs, 360 programs, and users at the PDP8 teletype console relatively painless.

For sending data to the 360 system:

ASYSMSG

PSYSMSG

For receiving data from the 360 system:

MSGES

LOAD

For communicating with the user via the teletype:

SYSMSG

SYSERR

The following facilities are provided by the UIMON8 system for the general convenience of all users:

1. High speed graphics time sharing terminal.
2. Relatively sophisticated text editor
    a. Text to teletype
    b. Text to dectape/disk
    c. Text to time sharing
3. PDP8 core image to time sharing file in reloadable format.

All user-oriented routines in the UIMON8 system assume a JMS type call with the 'IF' field set to 30 and the "DF' field set to the calling bank (i.e. the called routine will return to the user's program with the 'IF' and 'DF' fields set to the value of the 'DF' field at the time of the call).

The UIMON8 system resides entirely in bank 3 of the PDP8. Also, all of bank 1 is used as a buffer area when the text editor is being used and the first three words of bank 0 are used to link to the interrupt handler.

Locations 0 through 4777 of bank 3 contain the UIMON8 program. The area from 5000 through 5777 is reserved for user routines (page zero of bank 3 is completely filled with constants and save areas for the UIMON8 system; users may use the constants and pointers, but may not use any of the storage areas) and the area from 6000 through 6777 contains the character generator (part of the remaining area is "FREE" space to be used by a larger character generator with the last couple of pages in core being reserved for the resident routines of the future disk facilities).

The system interrupt handler is table driven, utilizing two tables. One with clear-flag or test-flag IOT's and the other with corresponding device routine addresses. Hence, to ignore interrupts from a particular device, it is merely necessary to change a test-flag IOT in the first table to a clear-flag IOT. Then, if an interrupt occurs from that device, the flag will automatically be cleared and a branch to the device routine will not be initiated. The converse procedure is used to accept interrupts from various devices. Clearly, if a user desired to handle interrupts from a particular device (such as the light pen) himself all that is necessary is for him to save the system routine address in the second table and substitute the address of him own routine. This is one of the main reasons for reserving part of bank 3 for user routines, i.e. for use by user's interrupt routines. Hence, each user who wishes to use any of the peripheral devices of the PDP8 need not construct his very own interrupt handler, but need only write routines to handle interrupts from devices for which system routines do not meet his special needs. (c.f. description of interrupt tables).

For communicating with the 360 system (the main purpose of UIMON8) two buffers for input and output, named respectively SBUFF and UBUFF, are maintained along with several flag words. A special "FILTER" routine (GMON8) looks at every input line from the 360 to see if it begins with a valid control character. Valid control characters cause the input line to be passed to a particular routine associated with each character. Hence, a form of message switching is obtained, e.g. information can be sent to various routines in the PDP8 intermitantly with no affect on one another. As an example, assume an interactive graphics program is running in the PDP8 with an analysis or interpreter program in the 360. If the 360 sent a line to the PDP8 with an invalid control character, the line (assumed to be a system message) would be automatically printed on the teletype and the user could type a coded response. (c.f. description of SYSERR routine) or the 360 program could have specified the control to call "LOAD", in which case the input line is assumed to be data in core load format. This data (being data directed in nature) could have replaced part or all of the current display file, set switches for the executing program, or could have consisted of merely input data to go into a user's own input buffer, or do anything else that a clever user might desire. Or, the 360 might have sent the control for entering the debugging package, or the 360 might have sent the control indicating that the input line was to go to the current input routine which might have been some other system routine or a particular routine of the user. All of the above input operations are performed with no effect on the currently executing program in the PDP8. In addition, since the control characters are associated with routines via a table, the user can add new ones or redefine old destinations, etc.

It is hopefully obvious at this point that input from the 360 to the PDP8 is handled practically on an automatic basis with the main constraint being that the input data is of the proper format. Hence, the major programming of input formatting winds up being done in the 360, where clearly programming is much easier for the average user (in addition when FORTRAN and PL/I interfaces are provided at the 360 end--work in progress--easy communication will exist for just about anyone).

For transmitting data to the 360, things are a bit more complex, but equally as flexible for the PDP8 user. To send core image data, i.e. data that may represent any bit pattern, the routine 'PSYSMSG' can be used. This routine can be called from any bank and will send a core block to the 360 as specified by the length, bank, and starting address information supplied in the calling parameters. Each PDP8 word is converted to two 8 bit characters to insure that no PDP7 control characters are accidentally transmitted, (c.f. description of data formats and PSYSMSG routine). To send ASCII character data to the 360, the routine 'ASYSMSG' is provided. This routine assumes that UBUFF contains a control character of the user's choice followed by the ASCII message (one character per word), followed by a carriage-return. The utility of this routine is made much clearer by the descriptions of the SYSMSG and MOVEDT routines.

ROUTINE: ASYSMSG

FUNCTION: Send a line of ASCII characters to the 360

CALLING SEQUENCE: ACC = Number of characters to be sent
(including a trailing carriage-return)

DESCRIPTION: The routine assumes that a line of ASCII characters with one character per word is already in UBUFF. This line is sent to the 360. If an error in transmission occurs, or if at some time during the transmission the 360 sends a message to the PDP8, this message will be printed on the teletype followed by the sequence 'CODE:'. If the user types a zero, the transmission operation will be terminated. However, any other character will cause the operation to be retried.

UBUFF may be filled by the user with the aid of the 'MOVEDT' routine or the SYSMSG facility (c.f. the descriptions of these routines). NOTE: No output line may exceed 125 (DEC) characters, including the carriage-return.

ROUTINE: PSYSMSG

FUNCTION: Send a line of core image data to the 360

CALLING SEQUENCE: ACC = Number of PDP8 words to be sent
PARAM1 = LOG-1 of the data
PARAM2 = CDF NN specifying the bank of the data

DESCRIPTION: The routine assumes that a user control character of some sort is in the first word of UBUFF. Each PDP8 word in the block to be sent is converted to two characters in the coded data format (c.f. description of data formats and is placed into UBUFF following the control character. Then a carriage-return is added, and the line is sent to the 360 in the same manner--and with the same error control--as with ASYSMSG).

NOTE: Since no output line may exceed 125(DEC) characters, the maximum number of words in the block is 61(DEC).

ROUTINE: MSGES (not user callable)

FUNCTION: Receive system messages and put into SBUFF

CALLING SEQUENCE: ACC = Last input character

DESCRIPTION: The routine sets the 'GO1' UIMON8 system flag immediately (thus inhibiting system routines from sending data to the 630), and sets the 'SYSFLG' when the entire message has been received. The input message will be printed on the current SBUFF printer when 'UPRIN' (the system buffer controller--not user callable--) or SYSERR (user callable) are called.

ROUTINE: LOAD

FUNCTION: Insert loader--formatted data from the 360 into PDP8 core

CALLING SEQUENCE: ACC = Starting address for data
PARAM = CDF NN specifying the bank

DESCRIPTION: The routine assumes that the input data is in loader format (c.f. data formats), hence, each input character is 6 bits of data or 6 bits of location information. The routine continues loading data until it receives an end-of-data character (211 octal). Since the input data can be of any amount and require more than one input line, carriage-returns and line feeds are ignored until the EOD character is received. The routine can be called explicitly by a user's program in anticipation of a data transmission from the 360. However, if the input is properly formatted, this routine will be automatically called by GMON8 (the remote-input filter routine). With the automatic-call feature, and with location information capabilities in the input stream, this routine provides an easy method for automatic data input, display file updating, switch setting, remote program loading, etc.

NOTE: This routine is really a load-and-go routine with a few special switches and options. When the EOD character is received, the routine checks the contents of its core address pointer. If the pointer is not zero, the routine uses that information plus the current bank information as the starting address of a program and branches to it. If the pointer was zero, the routine simply exits to the program that called it. Hence, all blocks of data that are sent through the loader must have the final three characters before the EOD character specifying new location information. This new location must either be the starting address of the program in the case of a program load, or must be zero in the case of a simple data load.

ROUTINE: SYSMSG

FUNCTION: Communicate fixed messages to the user via the teletype and format output lines from the user's responses.

CALLING SEQUENCE: ACC = N where N is a number > = Ø specifying a particular message residing in a message table defined by the user at system initialization.

DESCRIPTION: The routine assumes that a message table exists beginning at location 5000 in bank 3, i.e. at the beginning of the user's area. This table is placed into core by the user, either by a call to move the table from the user's program in another bank, by a call to dectape, or by a call to the disk routine. If a response is expected, the routine places the resulting teletype characters into UBUFF beginning at a location specified by the message table. Hence, by carefully constructing his table, the user can automatically format composite output lines from a particular sequence of related inquiries.

The table is formatted as follows:

```
SYSTBL -N              Expected return count
       UBUFF+M         ADDR of return characters in UBUFF
       MESSØ-1         ADDR-1 of message
       -L              Length of message in characters

       ...

       MESSØ           (Characters in packed format)
       MESS1

       ...

       MESSK
```

The expected return count includes a carriage-return that the user will type to end his response. So if a response of nine characters is expected, -12 would be specified. If the response that the user gave was only six characters instead of nine, the remaining character spaces in the UBUFF buffer would be blanked, and then the carriage-return would be inserted into UBUFF at the end of the nine character field.

If the user tried to type more than nine characters, the message would be retyped. If the user typed a percent sign, the message would also be retyped. If the expected return count is specified in the table as -1, no response is assumed, and the result is that only a message is sent to the teletype with no effect on UBUFF.

UBUFF+M (M> = Ø) specifies where in UBUFF the response of the user is to be put. NOTE: UBUFF can hold a maximum of 125(DEC) characters including carriage-return.

MESSØ-1 specifies the starting location of the message to be printed on the teletype.

-L specifies the length in characters of the message. If the expected return count was -1, the system adds a line feed and carriage-return to the line printed.

The table (K entries of four words each) is followed by the K variable length messages.

NOTE: The user could specify some other buffer area in bank 3 as the recipient of the teletype responses; however, if the user does this, it is his responsibility to insure that no part of the UIMON8 system is damaged.

ROUTINE: SYSERR

FUNCTION: Test the system buffers and return a code to the caller

CALLING SEQUENCE: None (ACC assumed = Ø)

DESCRIPTION: It is possible that the 360 or the PDP7 will at some time send a message to the PDP8 which is unrelated to the user's program, i.e. time sharing off, et. al. If UIMON8 is in time sharing or text editor mode at the time, these messages would automatically be printed on the teletype. However, in other modes (either system or user defined), immediate print-out would not necessarily occur, since printing is only done when a call to the system print routine is initiated. The complete arrival of a system message is indicated when UIMON8 sets its 'SYSFLG' to nonzero. When called, this routine prints the system buffers, and then tests 'SYSFLG'; if the flag was up (non-zero), the routine prints 'CODE:' and waits for a teletype reply. This reply (a number zero through seven) is returned to the calling program in the accumulator, and the 'SYSFLG' is reset to zero. Hence, the calling program has the option of taking various courses of action for any particular system message.

ROUTINE: MOVEDT

FUNCTION: Move a block of PDP8 core to anywhere in the machine

CALLING SEQUENCE: ACC = FØTØ (F=BANK FROM, T=BANK TO)
PARAM1 = ADDR FROM
PARAM2 = ADDR TO
PARAM3 = -COUNT (in words)

DESCRIPTION: The data at the specified 'FROM' address in the 'F' bank is moved sequentially starting at the lowest core address, and is moved to the 'TO' address in the 'T' bank. The user is cautioned about overlapping fields and should note that the action of this routine is identical to a 360 MVC instruction. This is merely a utility type routine and is included as a convenience for all users.

INTERRUPT TABLE DESCRIPTIONS:

Both TORCL and INTLST appear as follows at UIMON8 initialization, and the system assumes that the devices are in this order.

```
TORCL 6031          TEST KEYBOARD FLAG
      6041          TEST TELEPRINTER FLAG
      6631          TEST 630 REMOTE FLAG
      6132          TEST LIGHT PEN HIT FLAG
      7000          CLEAR MANUAL INTERRUPT FLAG
      7000          CLEAR EDGE FLAG
      6161          CLEAR EXTERNAL INTERRUPT
      6314          CLEAR INTERNAL INTERRUPT
      6764          CLEAR CLOCK INTERRUPT
      SZA           CLEAR DECTAPE FLAG
INTLST KEYTS        TEST FOR PUSH BUTTON HIT
      TPRTR         KEYBOARD
      GMON8         TELEPRINTER
      LPEN          REMOTE FILTER
      IRTRN         LIGHT PEN
      IRTRN         NO-OP
      IRTRN         NO-OP
      IRTRN         NO-OP
      IRTRN         NO-OP
      IRTRN         NO-OP
      IRTRN         NO-OP
      PBHIT?        PUSH BUTTON HIT
      PBHITS        IF PUSH BUTTON HIT, NO-OP
```

NOTE: If none of the other devices caused an interrupt, PBHIT? is entered to clear the flag, then PBHIT? calls PBHITS if the flag was on. Hence, user's should change PBHITS and not PBHIT?.

DATA FORMATS:

The UIMON8 system works with three types of data--ASCII, packed ASCII, and Loader formatted.

ASCII characters are stored one character per word in the right-most 8 bits. This data is always of a temporary nature, residing only in buffers just prior to being sent to the teletype or to the 360, or just after having been received from those devices.

Packed ASCII (really 6 bit trimmed ASCII corresponding identically to the right-most 6 bits of regular 8 bit ASCII characters) is used in the text editor display buffer, in all stored system messages, and is assumed to be the format of user messages stored in the user's area for the SYSMSG facility. When one of these messages is to be sent to the teletype, a system routine unpacks this data into UBUFF. Since the code is 6 bits, there are 64 valid parint characters available in stored messages or for display in the text editor, except that three of these are control characters for the text editor. These three are line feed (33), carriage-return (34), and escape data state (35), all of which should not be used by the user.

Loader formatted data is used to pass arbitrary bit patterns into and out of the PDP8. Since certain characters are control characters for the PDP7/360 system (such as carriage-return), UIMON8 must insure that such characters are not sent accidentally during the transmission of a data block. Hence, each PDP8 word of data is converted into two characters before being sent to the 360. Each character is of the form 1(6DATA BITS)0, with the top six bits of the PDP8 word going into the first character, and the right-most six going into the second character. Naturally, data coming into the system by way of the load routine is of the same form. The only other characters in this type are carriage-returns (215 octal), line feeds (212), EOD (211), and origin characters (20X, where X is 1, 3, 5, or 7 specifying banks 0 to 3, respectively). The only ambiguity in this setup is with line feed which could be=mistaken for a data character. However, line feeds are only sent by the PDP7 after a carriage-return at the end of each line, so by definition, 212 is only a line feed if it occurs after a carriage-return. Whenever an origin character occurs, the bank information is

extracted and used as the current load bank for the input data, and the next two data characters are assumed to contain a new starting address in that bank instead of being data.

SYSTEM LOCATIONS: (All locations in bank 3)

SYSTEM START('UTIL')---1000
ASYSMSG----------1600
PSYSMSG----------1614
LOAD----------1464
SYSMSG----------1656
SYSERR----------1731
MOVEDT----------3674
UBUFF----------3030
SBUFF----------3230
SYSFLG----------0007

The following routines may only be called by programs in bank 3, in the user's area, for example.

SCSET----------1211 (TS CONSOLE)
LD360----------1235 (LOAD 360 PARTITION)
LDPDP8----------1261 (LOAD FROM 360 INTO PDP8)
TEXTED----------1306 (TEXT EDITOR)
TERM8----------1327 (UIMON8 EXIT TO DECTAPE)

SYSTEM INTERRUPT TABLES:

TORCL----------1400 (TEST OR CLEAR IOT TABLE)
INTLST----------1413 (INTERRUPT ROUTINE LIST)

CHARACTER CODE EQUIVALENCES FOR THE COMMUNICATIONS SYSTEM MACROS:

STANDARD (EBCDIC)	TYPE A (ASCII)	TYPE B (CODED ASCII)	TYPE C (CODED BCD)
A	C1	82	E2
B	C2	84	E4
C	C3	86	E6
D	C4	88	E8
E	C5	8A	EA
F	C6	8C	EC
G	C7	8E	EF
H	C8	90	F0
I	C9	92	F2
J	CA	94	C2
K	CB	96	C4
L	CC	98	C6
M	CD	9A	C8
N	CE	9C	CA
O	CF	9E	CC
P	D0	A0	CE
Q	D1	A2	D0
R	D2	A4	D2
S	D3	A6	A4
T	D4	A8	A6
U	D5	AA	A8
V	D6	AC	AA
W	D7	AE	AC
X	D8	B0	AE
Y	D9	B2	B0
Z	DA	B4	B2
0	B0	E0	94
1	B1	E2	82
2	B2	E4	84
3	B3	E6	86
4	B4	E8	88
5	B5	EA	8A
6	B6	EC	8C
7	B7	EE	8E
8	B8	F0	90
9	B9	F2	92
NOT	87 BELL	87 BELL	FA BELL
UNDERSCORE	8A LF	UNUSED	FC LF
CENTS	89 EOD	89 EOD	89 EOD
PERCENT	A5	CA	FE
.	AE	DC	F6
<	BC	F8	BC
(	A8	D0	B8
+	AB	D6	E0
UPARROW	DE	BC	A0
AMPERSAND	A6	CC	80
!	A1	C2	9A
$	A4	C8	D6
*	AA	D4	D8

(CONTI.)

)	A9	D2	F8
;	BB	F6	BA
-	AD	DA	C0
/	AF	DE	A2
,	AC	D8	B6
>	BE	FC	BE
?	BF	FE	D4
:	BA	F4	9E
NUMSN	A3	C6	B4
EACH	C0	80	DA
'	A7	CE	98
=	BD	FA	96
"	A2	C4	9C
BLANK	A0	C0	80

ALL TYPES A, B, AND C ARE IN
HEX (I.E. A = 1010 BINARY)

BASIC JCL FOR ASSEMBLING 360 PROGRAMS WITH COMMUNICATIONS MACROS:

```
/*ID
//  EXEC ASM
//ASM.SYSLIB DD DSN=SYS1.MACLIB,DISP=OLD
//  DD DSN=SYS3.MACLIB,DISP=OLD
//  DD DSN=USER.GRAPHICS,MACLIB,DISP=OLD,UNIT=2314,    X
//          VOLUME=SER=UIRUR1
//ASM.SYSIN DD *
```

SOURCE DECK

```
/*
```

TO ADD THE OBJECT CODE OF PROPERLY ASSEMBLED PROGRAMS TO THE SYSTEM, CHANGE THE EXEC ASM CARD TO AN EXEC ASMLKED CARD AND AFTER THE /* ADD:

```
//LKED.SYSLMOD DD DSN=USER.GRAPHICS.LINKLIB(YOURNAME),    X
//         UNIT=2314,DISP=OLD,VOLUME=SER=UIRUR1
```

BASIC 360 JCL FOR ASSEMBLING PDP8 PROGRAMS WITH PDP8ASM:

```
/*ID
//JOBLIB DD UNIT=2314,VOLUME=SER=UIRUR1,    X
// EXEC PGM=PDP8ASM
//SYSUT1 DD UNIT=DRUM,SPACE=(TRK,(10,10)),DISP=NEW
//SYSUT8 DD DUMMY
//SYSUDUMP DD SYSOUT=A
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
```

SOURCE DECK

```
/*
```

TO PUT THE OBJECT CODE INTO THE MONITOR SYSTEM, CHANGE THE SYSUT8 CARD TO READ:

```
//SYSUT8 DD DSN=USER.GRAPHICS.LINKLIB(YOURNAME),    X
//           DISP=OLD,UNIT=2314,VOLUME=SER=UIRUR1
```

TO PUNCH THE OBJECT CODE ON A PAPER TAPE ON THE PDP7 IN LOADER FORMAT, CHANGE THE SYSUT8 CARD TO

```
//SYSUT8 DD UNIT=(CTC,,DEFER)
```

AND ADD THE FOLLOWING ASP CONTROL CARDS BEFORE THE JOBLIB CARD:

```
/*PROCESS MAIN
/*PROCESS FILE
/*FORMAL FI,DD=SYSUT8,DEST=PUNCH7,CONTROL=SINGLE
/*PROCESS PRINT
/*ENDPROCESS
```

Form AEC-427
(6/68)
AECM 3201

U. S. ATOMIC ENERGY COMMISSION
UNIVERSITY-TYPE CONTRACTOR'S RECOMMENDATION FOR DISPOSITION OF SCIENTIFIC AND TECHNICAL DOCUMENT

(See Instructions on Reverse Side)

1. AEC REPORT NO.
COO-1469-0114

2. TITLE USER PROGRAMMING SPECIFICATIONS FOR THE PDP8/360 COMMUNICATIONS SYSTEM

3. TYPE OF DOCUMENT (Check one):

- [X] a. Scientific end technical report
- [ ] b. Conference paper not to be published in a journal:
  - Title of conference _____
  - Date of conference _____
  - Exact location of conference _____
  - Sponsoring organizetion _____
- [ ] c. Other (Specify) _____

4. RECOMMENDED ANNOUNCEMENT AND DISTRIBUTION (Check one):

- [X] a. AEC's normal announcement and distribution procedures mey be followed.
- [ ] b. Make available only within AEC and to AEC contractors and other U.S. Government agencies end their contractors.
- [ ] c. Make no announcement or distrubution.

5. REASON FOR RECOMMENDED RESTRICTIONS:

6. SUBMITTED BY: NAME AND POSITION (Please print or type)

Dr. C. W. Gear
Professor of Computer Science and Applied Mathematics

Organization

Department of Computer Science
University of Illinois
Urbana, Illinois 61801

Signature C. W. Gear

Dete April 11, 1969

FOR AEC USE ONLY

AEC CONTRACT ADMINISTRATOR'S COMMENTS, IF ANY, ON ABOVE ANNOUNCEMENT AND DISTRIBUTION RECOMMENDATION:

PATENT CLEARANCE:

- [ ] a. AEC patent clearance has been granted by responsible AEC patent group.
- [ ] b. Report has been sent to responsible AEC patent group for clearance.
- [ ] c. Patent clearance not required.